ER622185268

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

SECURITY SCOPES AND PROFILES

INVENTORS:

GIOVANNI M. DELLA-LIBERA
VIJAY K. GAJJALA
TOMASZ JANCZUK
JOHN R. LAMBERT

ATTORNEY'S DOCKET NO. MS1-1857US

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to co-pending U.S. Patent Application [Attorney Docket No. MS1-1858US], filed February 16, 2004 entitled "Generic Security Claim Processing Model".

FIELD

[0002] Various embodiments described below relate generally to security systems for computing environments, and more particularly but not exclusively to security systems having a mechanism to identify types of information that need to be secured and another mechanism to specify how the types are to be secured.

BACKGROUND

[0003] Many message-based computing systems include security infrastructure to secure messages (e.g., integrity, confidentiality message authentication, signer authentication, sender authorization, etc.) sent from a sender to a recipient or receiver and to provide access control to resources targeted by the message. In some systems, a message requiring security can be sent to the receiver via one or more intermediate nodes. In a common scenario, an application to be run on a computing platform will have certain security requirements that the application attempts to satisfy via the security infrastructure.

[0004] Typical conventional security infrastructures are: (1) implemented either by a developer who writes the application (i.e., by which security is controlled by the code), which typically requires the developer to accurately know the environment in which the application will operate; or (2) configured/managed by an administrator for the computing system (after being installed by a deployer or configurer). Further, in typical conventional systems, the security infrastructure is selectively configured to either secure none of the messages or to secure all of the messages sent by the application. For example, the deployer can define a "policy" that all messages are to be secured using HyperText Transfer Protocol Secure (https).

SUMMARY

[0005] In accordance with aspects of the various described embodiments, a security system is provided with a mechanism to identify types of information that need to be secured and another mechanism to specify how the types are to be secured. The system includes a sender having an application and a receiver having a security module and one or more datastores to store information related to types of information that need to be secured (also referred to herein as "scopes"), how information is to be secured (also referred to herein as "profiles"), and a mapping between the scopes and profiles. In one embodiment, scopes not only identify the "type" of information to be secured, but also represent development time security decisions related to these "types" (e.g. whether it needs to be signed or encrypted). The profile compliments the scope by supplying deployment time decisions indicating how the requirements of the scope will be satisfied (e.g. what security tokens or algorithm to use). In one embodiment, scopes are implemented by application developers, whereas profiles are implemented by application deployers and/or administrators.

[0006] In operation, when a receiver receives the message, the receiver's security module determines which scope is appropriate for the message, and then determines the profile that is mapped to the scope via a binding. The security module can then make an access control decision using the profile.

[0007] This aspect allows more flexibility in implementing the security infrastructure (e.g., adjusting the security infrastructure to the particular environment, providing selectivity in which information is to be secured and how it is to be secured, and others). This aspect also simplifies the security tasks for the developer (e.g., the developer no longer needs to know the environment) and for the deployer and/or administrator (e.g., no need for access to source code, no need to provide sensitive information about users and/or the computing systems to the developer). In addition, the same application binary can be used in multiple deployment environments. Still further, scopes and profiles represent the security policy of the exchange, which can simplify the discovery and compliance with the security policy. In some embodiments, this discovery and compliance process can be automated.

[0008] In another aspect, a sender may also use scopes and profiles to secure an outgoing message. In a related aspect, intermediaries may also use scopes and profiles for security processing of messages. In some embodiments, these aspects need not be implemented.

[0009] In still another aspect, scopes are selected for a particular message using XPath-based expressions to determine whether a message has a preselected "pattern". In yet another aspect, scopes are selected by determining whether the message contains a preselected Simple Object Access Protocol (SOAP) action.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Non-limiting and non-exhaustive embodiments are described with reference to the following figures, wherein like reference numerals refer to like parts throughout the various views unless otherwise specified.

[0011] FIG. 1 is a block diagram illustrating a system with mechanisms to identify types of information that need to be secured and to specify how the message types are to be secured, which can be applied to senders, receivers and/or intermediaries, according to one embodiment.

[0012] FIG. 2 is a flow diagram illustrating operational flow in configuring a security module as depicted in the system of FIG. 1, according to one embodiment.

[0013] FIG. 3 is a flow diagram illustrating operational flow of a receiver as depicted in the system of FIG. 1 in security processing of a received message, according to one embodiment.

[0014] FIG. 4 is a flow diagram illustrating data flow in security processing of a received message, according to one embodiment.

[0015] FIG. 5 is a block diagram illustrating a sender as depicted in FIG. 1, according to one embodiment.

[0016] FIG. 6 is a block diagram illustrating an example computing environment suitable for practicing the above embodiments.

DETAILED DESCRIPTION

[0017] FIG. 1 is a block diagram illustrating a message-based system 100 with mechanisms to identify types of information that need to be secured and to specify how the message types are to be secured, according to one embodiment. In this example, system 100 includes a sender 102 and a receiver 104. Sender 102 and receiver 104, for example, can be different processes executing on a single computing platform, within the same process, or different nodes of a network. In general system 100 can include other nodes (not shown), which are omitted to enhance the clarity of this disclosure. Receiver 104 includes an application 112 and a security module 114. In accordance with this embodiment, receiver 104 also includes a scopes datastore 121, a bindings datastore 123 and a profiles datastore 125.

[0018] Scopes datastore 121 includes information (also referred to as "scopes") identifying types of information (e.g., messages) that need to be secured. In one embodiment, the developer(s) of application 112 define or provide scope(s) that each define a filter that, in effect, screen messages to identify those that are associated with that scope. Scopes also define general security settings required for messages that are identified with the scope. For example, a particular scope may require that the message be digitally signed and encrypted. In addition, scopes datastore 121 may include a default or "match-all" scope for messages that cannot be identified with any of the other scopes. Typically, scopes are representative of application-level, platform-independent and/or environment-independent message security settings.

[0019] In one example, a scope may be selected for a particular message using XPath-based expressions (e.g., based on XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999). In one embodiment, the XPath expressions are used to evaluate Boolean values (also referred to as "filters"). The XPath-based expressions can be used to determine whether a message has a preselected "pattern". The preselected pattern for a scope may be, for example, "//creditcard", which for some embodiments of application 112 defines a type of message (i.e., a message with credit card information) that must be encrypted and signed. In this example, the scope does not, however, specify how the information must be encrypted and signed. In general, the scope defines which parts of the message matching the scope need to be protected.

[0020] Alternatively, a scope may be selected for a particular message using an "action-based" scheme (e.g., based on an operation being requested by the message). For example, in a Simple Object Access Protocol-based (e.g., SOAP Version 1.2 Part 1: Messaging Framework W3C Recommendation 24 June 2003) embodiment, a scope may be selected by determining whether the message contains a preselected SOAP action (as defined by the SOAP Version 1.2 Standard or other related specification(s), such as WS-Addressing). In one embodiment, an XPath expression can be used to determine whether the message includes a particular SOAP action. The selected action may be, for example, "/action() = 'URN:bookstore:purchase", which for some embodiments of application 112 defines a type of message (i.e., a message that requests a purchase operation for purchases from a bookstore) that must be encrypted and signed. Again, in this example, the scope does not specify how the information must be encrypted and signed.

[0021] Profiles datastore 125 includes information (e.g., also referred to as "profiles") specifying particular settings/operations for securing messages. In one embodiment, the deployer(s) and/or administrator(s) of application 112 define or provide profile(s) that each define relatively more detailed (e.g., when compared to scopes) security settings (e.g., authentication, authorization, encryption/signature algorithms to be used in signing/encrypting a message) to be used with scope. For example, a particular profile may require that the message be digitally signed and encrypted using a selected username/password database (e.g., in a specified XML document) for authentication, and to use the 3DES algorithm for encryption. In contrast to scopes, profiles are typically representative of platform-dependent, environment-dependent and/or administrative-dependent message security settings.

[0022] Bindings datastore 123 includes information (e.g., also referred to herein as "bindings" or "mappings") specifying particular mappings of scope(s) to profile(s). In one embodiment, the deployer(s) and/or administrator(s) of application 112 map the scope(s) of scopes datastore 121 to the profile(s) of profiles datastore 125. In some embodiments, the deployer provides the mappings from scope(s) to profile(s) based on the deployer's knowledge of the profile(s), platform, environment, etc. In other embodiments, the developer may "hardcode" one or more selected bindings, depending on the use of application 112. For example, a developer of a medical application may "hardcode" bindings so that scopes related to patient medical information are always mapped to a profile that encrypts this confidential information. In addition, in some embodiments, bindings datastore 123 may include a default binding to map scopes that have not been explicitly bound to a profile. This default binding specifies a "fallback" profile. In this embodiment, the fallback profile, in effect, applies to all scopes but has a lower priority than an explicit binding.

[0023] FIG. 2 is a flow diagram illustrating operational flow in configuring scopes datastore 121 (FIG. 1), bindings datastore 123 (FIG. 1) and profiles datastores 125 (FIG. 1), according to one embodiment. In a block 202, one or more scopes are installed in scopes datastore 121. As previously described for one embodiment, along with application 112 (FIG. 1), developer(s) can define or implement scope(s) that are needed by or useful to application 112. The developer(s) can then, in this embodiment, provide the scope(s) to a system deployer or configurer. In this embodiment, the one or more of these scopes are installed by the system deployer.

[0024] In a block 204, one or more profiles are installed in profiles datastore 125. As previously described for one embodiment, the system deployer or configurer can define or specify one or more profiles in one embodiment. In this embodiment, the one or more profiles are installed by the deployer and can be maintained by the administrator.

[0025] In a block 206, the installed one or more scopes are mapped to one or more profiles. In one embodiment, each scope can only be mapped to one profile; however, multiple scopes can be bound to the same profile. As previously described for one embodiment, the system deployer or configurer can define or specify the mappings in one embodiment to be stored in bindings datastore 123. In this embodiment, the one or more mappings are installed by the deployer and can be maintained by the administrator.

[0026] In a further refinement, configuration information (e.g., profiles and bindings) can be shared between nodes (not shown) in system 100. For example, the administrator can synchronize profiles and bindings datastores across system 100. In another example, if system 100 includes network accessible storage (NAS), then profiles and bindings datastores 125 and 123 can reside in the NAS.

[0027] FIG. 3 is a flow diagram illustrating operational flow of receiver 104 as depicted in the system of FIG. 1 in security processing of a received message, according to one embodiment. Referring to FIGS. 1 and 3, this exemplary operational flow proceeds as set forth below.

[0028] In a block 302, a scope is selected for the received message. In this embodiment, security module 114 determines whether a scope of scopes datastore 121 matches the message. In one embodiment, security module 114 uses XPath-based expressions to determine whether the message is of a particular type associated with a scope contained in scopes datastore 121, as previously described. In another embodiment, security module 114 uses an action-based algorithm to determine whether the message matches a scope contained in scopes datastore 121, as described above. In other embodiments, different algorithms may be used to match a scope to the message.

[0029] In some scenarios, multiple scopes could apply to a single message (i.e., also referred to herein as overlapping scopes). In one embodiment, if this were to occur, the operation can be aborted and an error message generated. In another embodiment, security module 114 could be configured to determine at start-up or compile time whether two or more scopes could apply to one message. If so, security module 114 could then prompt for user intervention to refine the scopes so that the scopes no longer overlap. In an alternative embodiment, security module 114 could be configured to resolve overlapping scopes by merging the requirements of both scopes (if they are not conflicting). In yet another embodiment, one of the overlapping scopes could be selected based on some predetermined criteria (e.g., strictest security requirements, scopes could be assigned a ranking when defined, with the highest ranking scope being selected, etc.)

[0030] In a block 304, a profile is selected for the scope that was selected in block 302. In one embodiment, security module 114 uses a mapping contained in bindings datastore 123 to select a profile from profiles datastore 125. For example, in one embodiment, bindings datastore 123 includes a lookup table of profiles that are indexed by scope. In other embodiments, different schemes may be used to find a profile that is associated to the scope selected in block 302.

[0031] In a block 306, security requirement(s) derived from the scope selected in block 302 and the profile selected in block 304 are applied to the received message. In one embodiment, security module 114 applies the security requirements. For example, the security requirements can include authentication, authorization and access control requirements as described in the aforementioned co-pending U.S. Patent Application [Attorney Docket No. MS1-1858US] entitled "Generic Security Claim Processing Model".

[0032] In a decision block 308, it is determined whether the received message meets the security requirements that were applied in block 306. In one embodiment, security module 114 is configured to make this decision. If the security requirements are not met, in a block 310, the message is rejected. Conversely, if the security requirements are met, in a block 312, the message can undergo further processing. For example, the message can be processed by application 114.

[0033] FIG. 4 is a data flow diagram illustrating exemplary data flow 400 in security processing of a received message 403, according to one embodiment. Referring to FIGS. 1 and 4, data flows in one embodiment of system 100 as follows. Sender 102 sends message 403 to receiver 104. Security module 114 then performs a scope determination operation 405 on message 403. In this embodiment, in performing operation 405, security module 114 finds a scope from scope(s) 407 (e.g., stored in scopes datastore 121) that applies to the message, as indicated by an arrow 409. In one embodiment, security module 114 performs operation 405 as described in block 302 (FIG. 3).

[0034] Security module 114 then performs a profile determination operation 411 using the scope obtained from performing scope determination operation 405. In this embodiment, in performing operation 411, security module 114 finds a profile from profile(s) 417 (e.g., stored in profiles datastore 125) using a corresponding binding from bindings 413 (e.g., stored in bindings datastore 123), as indicated by arrows 415, 419 and 421. In one embodiment, security module 114 performs operation 411 as described in block 304 (FIG. 3).

[0035] The dataflow continues to an apply profile operation 423 using the profile obtained from performing apply profile operation 411. In one embodiment, security module 114 applies security requirements derived from the obtained profile to message 403 as described in block 306 (FIG. 3).

[0036] Security module 114 also performs access control operations to accept or deny message 403 based on whether message 403 met the security requirements of derived from the obtained scope and profile. In one embodiment, security module 114 makes an access control decision for message 403 as described in blocks 308, 310 and 312 (FIG. 3).

[0037] FIG. 5 illustrates an embodiment of sender 102 (FIG. 1), having optional "send-side" security scopes and profiles. In this example embodiment, sender 102 includes an application 512, a security module 514, and messaging infrastructure 517. In accordance with this embodiment, sender 102 also includes a scopes datastore 521, a bindings datastore 523 and a profiles datastore 525, which are substantially similar to datastores 121, 123 and 125 (FIG. 1), respectively.

[0038] In operation, a message to be sent by sender 102 would be processed by security module 514 before being passed to messaging infrastructure 517 for transmission to a receiver. In a manner similar to that described above (in conjunction with FIGS. 1 and 3) for security module 114, security module 514 would process the outgoing message to find a matching scope from scopes datastore 521, and map the matching scope to a corresponding profile from profiles datastore 525 using mappings from bindings datastore 523. The security requirements from the selected profile could include, for example, prohibiting unencrypted messages from being sent, or requiring a digital signature be added to the message. In some scenarios, portions of messages must remain unencrypted or if encrypted, using encryption methods encrypted with "lower" encryption algorithms to conform to applicable law. In such scenarios, this embodiment of sender 102 can be used to ensure that these message portions conform to the applicable law (or company policies or regulations for workplace scenarios).

[0039] In a further refinement, this embodiment of sender 102 can be configured to obtain the receiver's security requirements or "policies". For example, a receiver may have a policy that all messages must be secured using specified mechanisms/methods. In yet another refinement, in scenarios in which a node in system 100 (FIG. 1) may serve as both a sender and a receiver, the node would have both send-side and receive-side scopes/profiles/bindings that would be used depending on whether the node was sending or receiving a message.

[0040] The various embodiments described above may be implemented in computer environments of the senders and receivers. An example computer environment suitable for use in the senders and receivers is described below in conjunction with FIG. 6.

[0041] FIG. 6 illustrates a general computer environment 600, which can be used to implement the techniques described herein. The computer environment 600 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computer environment 600 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the example computer environment 600.

[0042] Computer environment 600 includes a general-purpose computing device in the form of a computer 602. The components of computer 602 can include, but are not limited to, one or more processors or processing units 604, system memory 606, and system bus 608 that couples various system components including processor 604 to system memory 606.

[0043] System bus 608 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include an Industry Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) local bus, a Peripheral Component

Interconnects (PCI) bus also known as a Mezzanine bus, a PCI Express bus, a Universal Serial Bus (USB), a Secure Digital (SD) bus, or an IEEE 1394, *i.e.*, FireWire, bus.

[0044] Computer 602 may include a variety of computer readable media. Such media can be any available media that is accessible by computer 602 and includes both volatile and non-volatile media, removable and non-removable media.

[0045] System memory 606 includes computer readable media in the form of volatile memory, such as random access memory (RAM) 610; and/or non-volatile memory, such as read only memory (ROM) 612 or flash RAM. Basic input/output system (BIOS) 614, containing the basic routines that help to transfer information between elements within computer 602, such as during start-up, is stored in ROM 612 or flash RAM. RAM 610 typically contains data and/or program modules that are immediately accessible to and/or presently operated on by processing unit 604.

[0046] Computer 602 may also include other removable/non-removable, volatile/non-volatile computer storage media. By way of example, FIG. 6 illustrates hard disk drive 616 for reading from and writing to a non-removable, non-volatile magnetic media (not shown), magnetic disk drive 618 for reading from and writing to removable, non-volatile magnetic disk 620 (*e.g.*, a "floppy disk"), and optical disk drive 622 for reading from and/or writing to a removable, non-volatile optical disk 624 such as a CD-ROM, DVD-ROM, or other optical media. Hard disk drive 616, magnetic disk drive 618, and optical disk drive 622 are each connected to system bus 608 by one or more data media interfaces 625. Alternatively, hard disk drive 616, magnetic disk drive 618, and optical disk drive 622 can be connected to the system bus 608 by one or more interfaces (not shown).

[0047] The disk drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules, and other data for computer 602. Although the example illustrates a hard disk 616, removable magnetic disk 620, and removable optical disk 624, it is appreciated that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes or other magnetic storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or other optical storage, random access memories (RAM), read only memories (ROM), electrically erasable programmable read-only memory (EEPROM), and the like, can also be utilized to implement the example computing system and environment.

[0048] Any number of program modules can be stored on hard disk 616, magnetic disk 620, optical disk 624, ROM 612, and/or RAM 610, including by way of example, operating system 626, one or more application programs 628, other program modules 630, and program data 632. Each of such operating system 626, one or more application programs 628, other program modules 630, and program data 632 (or some combination thereof) may implement all or part of the resident components that support the distributed file system.

[0049] A user can enter commands and information into computer 602 via input devices such as keyboard 634 and a pointing device 636 (*e.g.*, a "mouse"). Other input devices 638 (not shown specifically) may include a microphone, joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and other input devices are connected to processing unit 604 via input/output interfaces 640 that are coupled to system bus 608, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB).

[0050] Monitor 642 or other type of display device can also be connected to the system bus 608 via an interface, such as video adapter 644. In addition to monitor 642, other output peripheral devices can include components such as speakers (not shown) and printer 646, which can be connected to computer 602 via I/O interfaces 640.

[0051] Computer 602 can operate in a networked environment using logical connections to one or more remote computers, such as remote computing device 648. By way of example, remote computing device 648 can be a PC, portable computer, a server, a router, a network computer, a peer device or other common network node, and the like. Remote computing device 648 is illustrated as a portable computer that can include many or all of the elements and features described herein relative to computer 602. Alternatively, computer 602 can operate in a non-networked environment as well.

[0052] Logical connections between computer 602 and remote computer 648 are depicted as a local area network (LAN) 650 and a general wide area network (WAN) 652. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

[0053] When implemented in a LAN networking environment, computer 602 is connected to local network 650 via network interface or adapter 654. When implemented in a WAN networking environment, computer 602 typically includes modem 656 or other means for establishing communications over wide network 652. Modem 656, which can be internal or external to computer 602, can be connected to system bus 608 via I/O interfaces 640 or other appropriate mechanisms. It is to be appreciated that the illustrated network connections are examples and that other means of establishing at least one communication link between computers 602 and 648 can be employed.

[0054] In a networked environment, such as that illustrated with computing environment 600, program modules depicted relative to computer 602, or portions thereof, may be stored in a remote memory storage device. By way of example, remote application programs 658 reside on a memory device of remote computer 648. For purposes of illustration, applications or programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of computing device 602, and are executed by at least one data processor of the computer.

[0055] Various modules and techniques may be described herein in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. for performing particular tasks or implement particular abstract data types. These program modules and the like may be executed as native code or may be downloaded and executed, such as in a virtual machine or other just-in-time compilation execution environment. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.

[0056] An implementation of these modules and techniques may be stored on or transmitted across some form of computer readable media. Computer readable media can be any available media that can be accessed by a computer. By way of example, and not limitation, computer readable media may comprise "computer storage media" and "communications media."

[0057] "Computer storage media" includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by a computer.

[0058] "Communication media" typically embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as carrier wave or other transport mechanism. Communication media also includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. As a non-limiting example only, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above are also included within the scope of computer readable media.

[0059] Reference has been made throughout this specification to "one embodiment," "an embodiment," or "an example embodiment" meaning that a particular described feature, structure, or characteristic is included in at least one embodiment of the present invention. Thus, usage of such phrases may refer to more than just one embodiment. Furthermore, the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

[0060] One skilled in the relevant art may recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, resources, materials, *etc*. In other instances, well known structures, resources, or operations have not been shown or described in detail merely to avoid obscuring aspects of the invention.

[0061] While example embodiments and applications have been illustrated and described, it is to be understood that the invention is not limited to the precise configuration and resources described above. Various modifications, changes, and variations apparent to those skilled in the art may be made in the arrangement, operation, and details of the methods and systems of the present invention disclosed herein without departing from the scope of the claimed invention.